

2021.06.18. 교육

나정휘

<https://justicehui.github.io/>

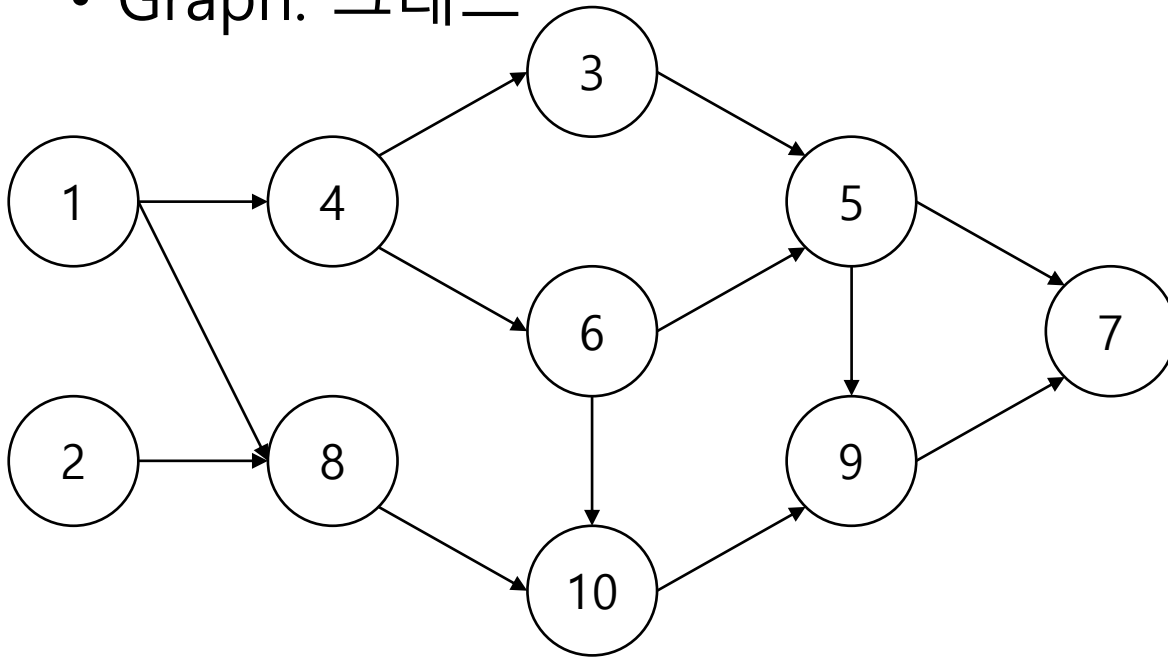
목차

- 위상 정렬
- DP
- USACO 8문제 풀이

위상 정렬

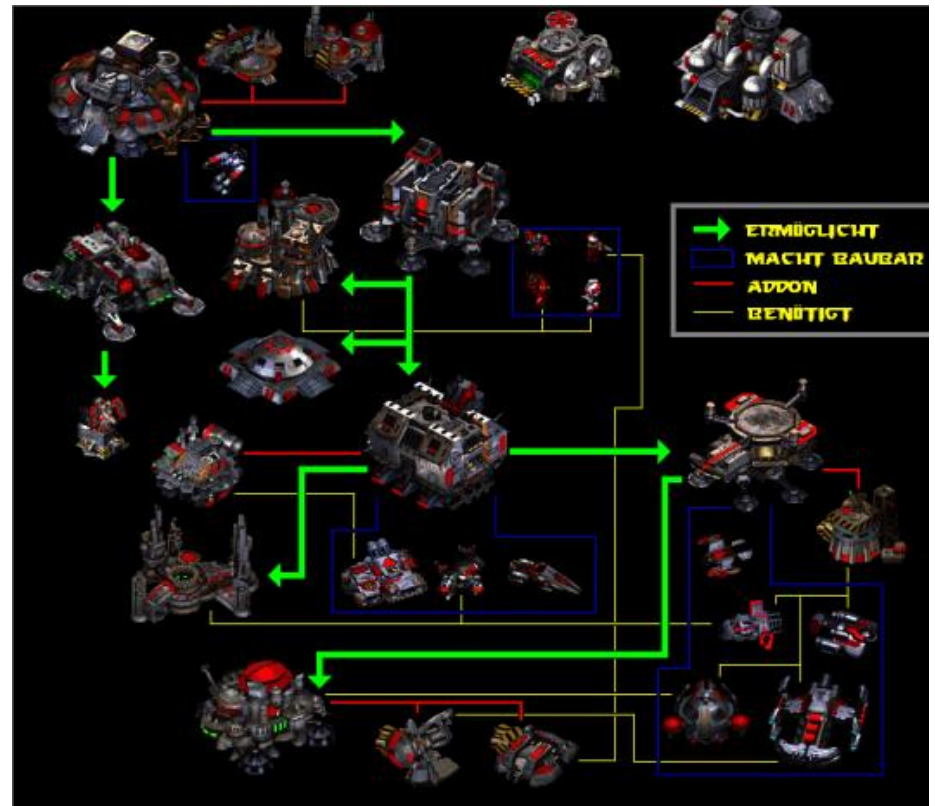
DAG란?

- Directed Acyclic Graph
 - Directed: 유향, 간선에 방향이 정해져 있는
 - Acyclic: 사이클이 없는
 - Graph: 그래프



위상 정렬

- 정점들을 간선의 방향을 거스르지 않는 순서대로 나열하는 것
 - $A \rightarrow B$ 간선이 있다면 A는 B보다 먼저 나와야 함
 - 위상 정렬 순서대로 나열했을 때, 간선은 모두 오른쪽으로 향하게 됨
- Valid한 건물 건설 순서 구하기
- 여러 가지가 나올 수 있음



어떻게 구하지?

- BFS를 응용하는 방법
- DFS를 응용하는 방법
- 혹시 BFS/DFS 모르는 사람?

위상 정렬의 성질

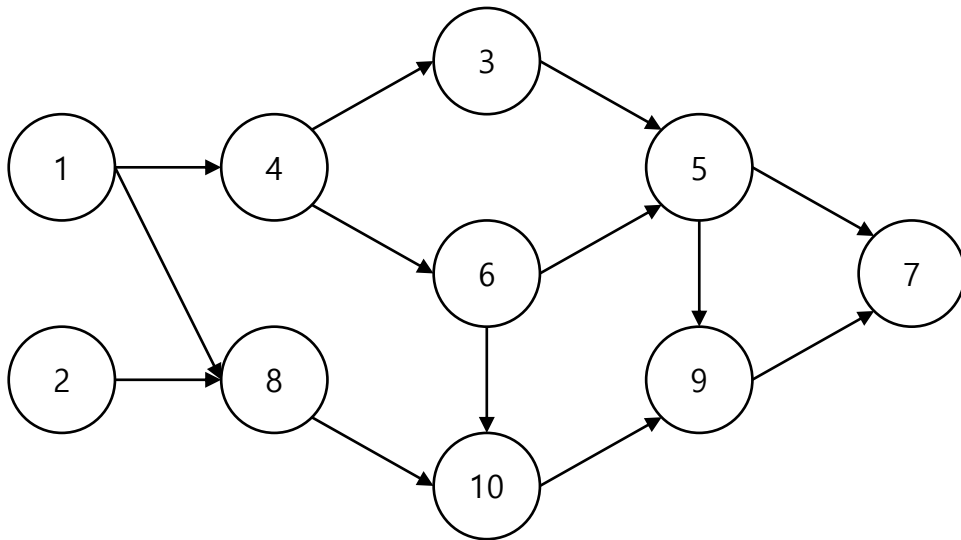
- 가장 앞에 오는 정점은 항상 in-degree가 0임
 - Why?
- in-degree가 0이면서 맨 앞이 아닌 정점은 앞으로 옮겨도 됨
 - Why?

질문?

BFS를 응용하는 방법

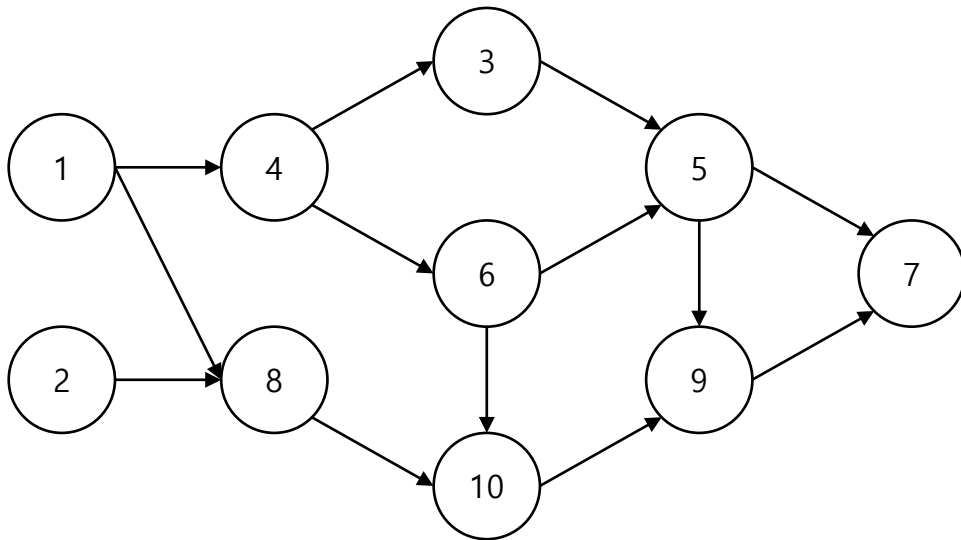
- in-degree가 0인 정점을 Queue에 삽입
- Queue가 비어 있지 않으면 아래 과정을 반복
 - 큐에서 정점 v 를 제거
 - v 를 정답에 추가
 - v 에서 뺀어 나가는 모든 간선을 제거, 해당 정점 in-degree 1 감소
 - in-degree가 0이 된 정점을 Queue에 삽입
- 모든 정점을 방문하지 않고 종료됐다면 사이클 있는 그래프
 - Why?

예시



- $Q = \{1, 2\}$
- $V = 1$ $Q = \{2, 4\}$
- $V = 2$ $Q = \{4, 8\}$
- $V = 4$ $Q = \{8, 3, 6\}$
- $V = 8$ $Q = \{3, 6\}$
- $V = 3$ $Q = \{6\}$
- $V = 6$ $Q = \{10, 5\}$
- $V = 10$ $Q = \{5\}$
- $V = 5$ $Q = \{9\}$
- $V = 9$ $Q = \{7\}$
- $V = 7$ $Q = \{\}$
- 1 2 4 8 3 6 10 5 9 7

예시



- $Q = \{2, 1\}$
- $V = 2$ $Q = \{1\}$
- $V = 1$ $Q = \{8, 4\}$
- $V = 8$ $Q = \{4\}$
- $V = 4$ $Q = \{6, 3\}$
- $V = 6$ $Q = \{3, 10\}$
- $V = 3$ $Q = \{10, 5\}$
- $V = 10$ $Q = \{5, 9\}$
- $V = 5$ $Q = \{9\}$
- $V = 9$ $Q = \{7\}$
- $V = 7$ $Q = \{\}$
- 2 1 8 4 6 3 10 5 9 7

시간 복잡도

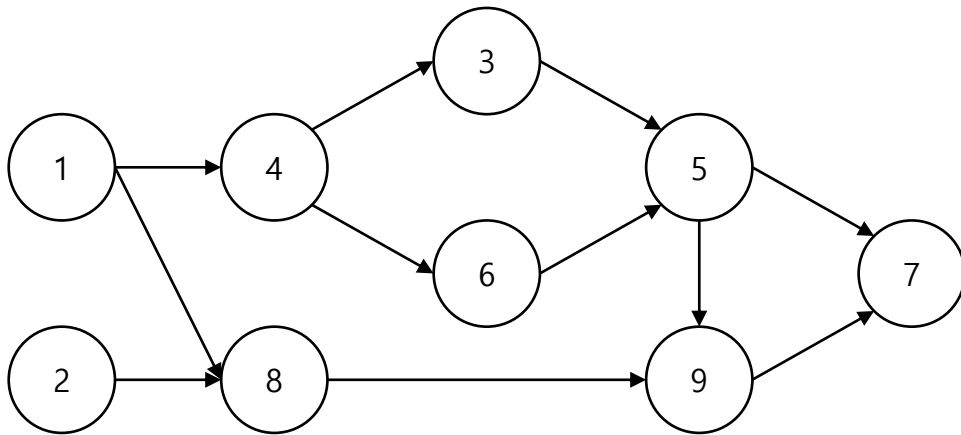
- 정점 N 개, 간선 M 개
- $O(N+M)$

질문?

DFS를 응용하는 방법

- DFS를 하면서, 빠져나오는 순서대로 기록한다.
- 빠져나온 순서의 역순은 위상 정렬 순서이다.
- 왜 그럴까?
 - 간선 $A \rightarrow B$ 가 있을 때
 - A를 먼저 방문하면 항상 B를 먼저 빠져나오는가?
 - B를 먼저 방문하면 항상 B를 먼저 빠져나오는가?

예시



- 1 진입
 - 4 진입
 - 6 진입
 - 5 진입
 - 7 진입
 - 7 탈출(5)
 - 9 진입
 - 9 탈출(5)
 - 5 탈출(6)
 - 7 9 5 6 3 4 8 1 2
 - 2 1 8 4 3 6 5 9 7
- 6 탈출(4)
 - 3 진입
 - 3 탈출(4)
 - 4 탈출(1)
 - 8 진입
 - 8 탈출(1)
 - 1 탈출(x)
 - 2 진입
 - 2 탈출(x)

시간 복잡도

- 정점 N 개, 간선 M 개
- $O(N+M)$

질문?

동적 계획법

동적 계획법

- 동적 계획법
 - 복잡한 문제를 간단한 문제들로 나누고
 - 간단한 문제들을 해결한 뒤
 - 간단한 문제들의 답을 이용해 복잡한 문제의 답을 구함
- 떠오르는 질문 리스트
 - 문제의 복잡성이 뭐지?
 - 큰 문제 / 작은 문제로 생각하면 편함
 - 항상 가능하지는 않을텐데...
 - 최적 부분 구조 (Optimal Substructure)
 - 이게 효율적일까?
 - 중복되는 부분 문제 (Overlapping Subproblem)

최적 부분 구조 (Optimal Substructure)

- 큰 문제의 최적해가 작은 문제의 최적해를 포함한다.
 - 성립하지 않으면 작은 문제의 답을 이용해서 큰 문제의 답을 구할 수 없음
- ex) 피보나치 수열
 - 큰 문제: fibonacci(N)
 - 작은 문제: fibonacci(0), fibonacci(1)
 - fibonacci(N)은 $a \cdot \text{fibonacci}(0) + b \cdot \text{fibonacci}(1)$ 로 나타낼 수 있다.

Overlapping Subproblem

- 작은 문제의 답을 여러 번 참조해야 한다.
 - 한 번 계산한 답을 **저장**해두면, 다시 참조할 때 연산량을 줄일 수 있음
- ex) 피보나치 수열
 - 큰 문제: fibonacci(N), $N > 5$
 - 작은 문제: fibonacci(5), fibonacci(4), fibonacci(3), ...
 - fibonacci(N)은 $a \cdot \text{fibonacci}(5) + b \cdot \text{fibonacci}(4)$ 로 나타낼 수 있다
 - 이때 fibonacci(5), fibonacci(4)를 각각 a, b번 호출할텐데
 - 한 번 계산한 다음에 답을 저장하면 실행 시간을 크게 줄일 수 있음

질문?

주어진 문제를 DP로 풀 수 있다는 것을 어떻게 알 수 있나요?

이게 동적 계획법 문제인가?

- 동적 계획법 문제의 특징
 - 큰 문제를 **한 개 이상의 작은 문제**로 분할할 수 있어야 함
 - 큰 문제와 작은 문제를 **동일한 방법**으로 풀 수 있어야 함 (귀납법)
 - 큰 문제의 최적해가 작은 문제의 최적해들로 구성되어야 함
- 결론: Optimal Substructure, Overlapping Subproblem을 알아야 함

이게 동적 계획법 문제인가?

- 동적 계획법 문제인지 판단하는 방법
 - **Optimal Substructure 성질을 만족함을 증명한다.**
 - N년 간 문제를 풀어본 경험을 토대로 추측한다.
 - 웬지 DP일 것 같으니까 일단 믿어본다.
- 증명할 자신이 없으면 그냥 **문제를 많이 풀어서** 유형을 외우자.
- solved.ac 기준 골드까지는 물량으로 밀 수 있다.

동적 계획법을 써야하는 건 알겠는데...

- 큰 문제와 작은 문제 간의 상관관계(점화식)을 어떻게 찾지?
 - 점화식을 정의한다.
 - 현재 "상태"를 잘 표현할 방법을 생각해본다.
 - 배열 인덱스, 선택한 원소의 개수, 선택한 원소의 합/곱(을 나눈 나머지) 등
 - (최적화) 상태의 개수를 줄여도 온전하게 표현할 수 있는지 생각해본다.
 - (최적화) 다른 방식으로 표현해볼 수는 없을까?
 - 점화 관계를 찾는다.
 - 현재 "상태"로 가기 바로 직전 "상태"는 무엇이 있을까?
 - (최적화) 다양한 DP 최적화들(CHT, DnC Opt, Kitamasa 등)

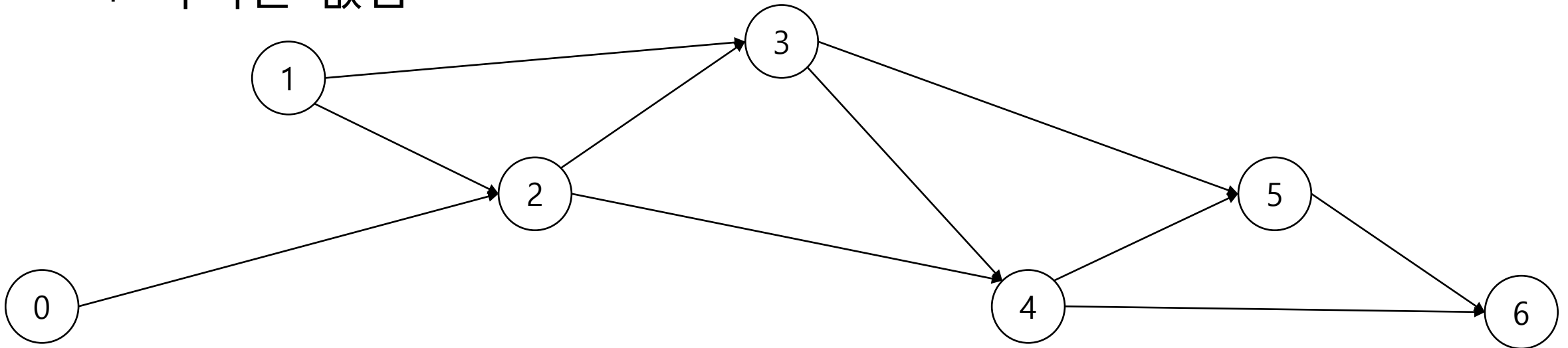
질문?

DAG와 DP의 관계

- DAG(Directed Acyclic Graph): 사이클 없는 방향 그래프
 - 사이클이 없다
 - 위상 정렬을 할 수 있다
- DP: 작은 문제의 답을 이용해 큰 문제의 답을 구하는 방법
 - 작은 문제와 큰 문제 간의 연결 관계?

DAG와 DP의 관계 - 예시

- 피보나치 수열
- 위상 정렬 순서가 앞선 것의 답을 알아야 뒤에 있는 것의 답을 구할 수 있음
+ 사이클 없음



DAG와 DP의 관계

- 어떤 문제는 DAG로 생각해서 푸는 것이 편할 수도 있다.
 - ex) 최솟값 <-> 그래프 최단 경로
 - 몇몇 논문을 보면 DP를 평면 그래프로 생각해서 분할 정복을 하던데...
- 사이클이 없는 방향 그래프에서 경로의 개수는 DP로 해결
 - Why?

더 공부할 거리

- 선형 점화식의 최적화
 - 행렬 거듭제곱
 - Kitamasa Method (+ FFT)
- 함수의 개형(특히 기울기)를 이용한 DP 최적화
 - Stack/Sqrt Decomposition/BBST를 활용한 Convex Hull Trick
 - Li-chao Tree
 - Slope Trick
- Monge Array의 성질
 - Divide and Conquer Optimization
 - Monotone Queue Optimization
 - Aliens Trick
- 다 필요 없고 Gold V ~ Platinum V에 있는 기초 DP나 잘 풀자.

USACO 문제 목록

• 20650	2020 Dec B1	Do You Know Your ABCs	Bronze II
• 18787	2020 Feb B2	Mad Scientist	Silver V
• 20647	2020 Dec S1	Cowntagion	Gold IV
• 18780	2020 Feb G1	Timeline	Gold II
• 18879	2020 Opn S3	The Moo Particle	Gold I
• 20648	2020 Dec S2	Rectangular Pasture	Platinum V
• 20649	2020 Dec S3	Stuck in a Rut	Platinum V
• 18874	2020 Opn G1	Haircut	Platinum IV

20650. Do You Know Your ABCs

USACO 2020 December Bronze 1번

문제 요약

- $0 < A \leq B \leq C$ 를 만족하는 세 정수 A, B, C 가 있을 때
- $A, B, C, A+B, B+C, C+A, A+B+C$ 가 무작위 순서로 주어짐
- A, B, C 를 각각 구하는 문제

풀이

- 입력으로 주어지는 7개의 자연수 중
 - 가장 큰 값은 $A+B+C$
 - 두 번째로 큰 값은 $B+C$
 - 세 번째로 큰 값은 $A+C$
- $A = (A+B+C) - (B+C)$
- $B = (A+B+C) - (A+C)$
- $C = (A+B+C) - A - B$

18787. Mad Scientist

USACO 2020 February Bronze 2번

문제 요약

- 'G', 'H'로 이루어진 문자열 A, B가 주어짐
- 아래와 같은 연산을 0번 이상 수행해서 B를 A로 만들어야 함
 - B의 부분 문자열을 선택해서 'G'는 'H'로, 'H'는 'G'로 변경
- 연산의 최소 횟수를 구하는 문제

풀이

- $C[i]$ 를 $A[i]$ 와 $B[i]$ 가 같으면 0, 다르면 1이라고 정의하자.
- $C[i]$ 에 등장하는 "1의 덩어리의 개수"를 구하는 문제가 된다.

```
int ans = 0, flag = 0;
for(int i=1; i≤N; i++){
    if(C[i]) flag = 1;
    else ans += flag, flag = 0;
}
ans += flag;
cout << ans;
```

20647. Cowntagion

USACO 2020 December Silver 1번

문제 요약

- N개의 목장이 도로를 이용해 트리 형태로 연결되어 있음
- 1번 목장에 있는 소 한 마리가 COWVID-19에 감염됨
- 매일 아래 두 가지 사건 중 한 가지가 발생함
 - 한 목장에 있는 확진자의 수가 2배가 됨
 - COWVID-19에 감염된 소 한 마리가 도로를 통해 이동함
- N개의 목장에 모두 COWVID-19에 감염된 소가 존재하게 되는 최소 시간을 구하는 문제

풀이

- 풀이가 잘 보이지 않을 때는 쉬운 문제로 바뀌어서 생각해보자.
 - 성계 그래프
 - 선형 트리
- 성계 그래프의 경우
 - 소가 N 마리 이상이 될 때까지 기다린 다음, 한 마리 씩 이동
 - $N - 1 + \lceil \log_2 N \rceil$
- 선형 트리의 경우
 - 2마리로 만들고 아래로 보내는 것을 반복
 - $2(N - 1)$

풀이

- 두 가지 풀이를 조합하면 일반적인 트리에서 문제를 풀 수 있다.
 - 각 정점에서 소가 (자식 정점의 수+1)마리 이상이 될 때까지 기다린 뒤
 - 한 마리 씩 이동하면 된다.
- $\lceil \log_2(C + 1) \rceil$ 을 구하는 방법에 따라 시간 복잡도가 달라지는데
- 아무리 느려도 $O(N \log N)$ 에는 풀 수 있다.

18780. Timeline

USACO 2020 February Gold 1번

문제 요약

- M일 동안 N개의 세션이 진행된다.
- 입력으로 S_1, S_2, \dots, S_N 과 (a, b, x) 꼴의 튜플 C개가 주어진다.
- 아래 조건을 만족하도록 세션을 열어야 한다.
 - i번 세션은 S_i 일 이후에 열려야 한다.
 - b번 세션은 a번 세션보다 최소 x일 이상 늦게 열린다.
- 각 세션이 열릴 수 있는 가장 빠른 날짜를 구하는 문제

풀이

- a가 열리고 x일 지난 후에 b가 열릴 수 있다.
 - a에서 b로 가는 가중치가 x인 간선
 - DAG에서 최장 경로를 구하는 문제
- S_i
 - 0일차에 시작하는 0번 세션을 만든 뒤
 - $(0, i, S_i)$ 튜플 추가
- $O(N+C)$ 에 풀 수 있음

18879. The Moo Particle

USACO 2020 US Open Silver 3번

문제 요약

- 2차원 평면에 N 개의 점 (x_i, y_i) 가 주어진다.
- 두 점 간의 "상호 작용"은 아래와 같이 정의된다.
 - 만약 두 점 i, j 가 $x_i \leq x_j$ & $y_i \leq y_j$ 를 만족하면
 - 두 점 중 한 점이 없어진다.
- 여러 번의 "상호 작용"을 거쳐 점의 개수가 감소할 수 있는데,
- 이때 남는 점의 개수의 최솟값을 구하는 문제

풀이

- 두 점 i, j 의 "상호 작용"을 i 와 j 를 간선으로 잇는다고 생각하자.
- 간선을 모두 만든 뒤, 컴포넌트의 개수가 문제의 정답이 된다.
- 좌표 대소 비교를 하는 문제는 보통 한 축으로 정렬하면 편하다.
- x 좌표 기준으로 정렬하고 생각하자.

풀이

- x좌표 기준으로 정렬하자.
 - $1 \leq j < i$ 번째 점은 $y_j \leq y_i$ 이면 i 번째 점과 "상호 작용"할 수 있다.
 - $i < j \leq N$ 번째 점은 $y_j \geq y_i$ 이면 i 번째 점과 "상호 작용"할 수 있다.
- i 번째 점과 $i+1$ 번째 점이 같은 컴포넌트에 속하지 않을 조건은
 - $\min(y_1, \dots, y_i) > \max(y_{i+1}, \dots, y_n)$
- 정답 = (이 조건을 만족하는 i 의 개수) + 1

20648. Rectangular Pasture

USACO 2020 December Silver 2번

문제 요약

- 목장에 N 마리의 소가 있다.
- 소들의 부분 집합은 2^N 가지가 존재하는데
- 변이 x, y 축에 평행한 직사각형으로 커버할 수 있는 부분 집합의 개수를 구하는 문제

풀이

- 좌표 범위가 10억인 건 보기 싫으니까 좌표 압축을 하자.
 - x, y 좌표가 각각 서로 다르기 때문에 별 생각 없이 해도 된다.
- 두 가지 방법을 생각해볼 수 있다.
 - 가능한 부분 집합의 개수를 직접 세는 경우
 - 전체 경우의 수에서 불가능한 집합의 개수를 빼는 경우
- 2^{2500} 같은 큰 수는 C++에서 처리하기 어렵기 때문에
- 전자가 좋을 것 같다는 생각을 할 수 있다.

풀이

- x, y 좌표는 모두 1 이상 N 이하다.
- 직사각형의 왼쪽/오른쪽 변을 고정하자: $O(N^2)$ 가지
 - 왼쪽 변 : (x_i, y_i) , 오른쪽 변 : (x_j, y_j)
 - 가능한 위/아래 변의 조합의 수를 구하면 문제를 해결할 수 있다.
 - (가능한 윗변의 개수) \times (가능한 아랫변의 개수)

풀이

- (x, y) 가 윗변이 될 수 있는 조건
 - $x_i \leq x \leq x_j$
 - $\max(y_i, y_j) \leq y$
- (x, y) 가 아랫변이 될 수 있는 조건
 - $x_i \leq x \leq x_j$
 - $y \leq \min(y_i, y_j)$

풀이

- 2차원 영역에서 어떤 직사각형에 속하는 점의 개수를 구해야 함
 - 2D Prefix Sum : 전처리 $O(N^2)$, 쿼리 $O(1)$
- 왼쪽/오른쪽 변 고정하면 $O(1)$ 에 구할 수 있음
- 전체 시간 복잡도는 $O(N^2)$
- 공집합 / 크기가 1인 집합 조심

20649. Stuck in a Rut

USACO 2020 December Silver 3번

문제 요약

- 위로 뺄어나가는 직선과 오른쪽으로 뺄어나가는 직선이 있음
- 각 직선은 다른 직선의 자취에 부딪히면 죽음
- 각 직선이 죽는 시점을 구하는 문제

풀이

- 위로 뺀는 직선들만 남겨두자.
- 오른쪽으로 뺀는 직선을 하나씩 보면서
- 위로 뺀는 직선이 죽는지, 오른쪽으로 뺀는 직선이 죽는지 확인
- $O(N)$ 스위핑을 N 번 수행 : $O(N^2)$

18874. Haircut

USACO 2020 US Open Gold 1번

문제 요약

- 길이 N 인 배열 A 가 주어짐
- $0 \leq j < N$ 인 자연수 j 에 대해
- j 보다 큰 수를 모두 j 로 바꿨을 때의 inversion 개수 구하는 문제

풀이

- 목표: $x < y \ \& \ A[x] > A[y] \ \& \ A[y] < j$ 인 (x, y) 순서쌍의 개수
- y 에 대해 $x < y \ \& \ A[x] > A[y]$ 를 만족하는 x 의 개수는 쉬움
 - Fenwick Tree / Segment Tree 등으로 구할 수 있음
- $R[j]$ 를 $A[y] = j$ 인 y 들에 대한 x 의 개수라고 정의하자.
- 문제의 정답은 $R[0] + R[1] + \dots + R[j-1]$ 이 된다.
- $O(N \log N)$ 에 풀 수 있다.